BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

Patent
Attorney Docket No. 032881-004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of
Michael John SYKES et al.
Application No.: 10/677,297
Filing Date: October 3, 2003
Title: ADAPTIVELY INTERFACING WITH A DATA REPOSITORY

Group Art Unit: 2171
Examiner: Unassigned
Confirmation No.: 9662

SUBMISSION OF CERTIFIED COPY OF PRIORITY DOCUMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

The benefit of the filing date of the following priority foreign application(s) in the following foreign country is hereby requested, and the right of priority provided in 35 U.S.C. § 119 is hereby claimed.

Country: Australia

Patent Application No(s).: 2002951909

Filed: October 4, 2002

In support of this claim, enclosed is a certified copy(ies) of said foreign application(s). Said prior foreign application(s) is referred to in the oath or declaration. Acknowledgment of receipt of the certified copy(ies) is requested.

Respectfully submitted,

BURNS, DOANE, SWECKER & MATHIS, L.L.P.

By _____
James A. LaBarre
Registration No. 28,632

P.O. Box 1404
Alexandria, Virginia 22313-1404
(703) 836-6620

Date: March 8, 2004

Patent Office
Canberra

I, JONNE YABSLEY, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. 2002951909 for a patent by TENIX INDUSTRIES PTY LIMITED as filed on 04 October 2002.

WITNESS my hand this
Tenth day of October 2003

JONNE YABSLEY
TEAM LEADER EXAMINATION
SUPPORT AND SALES

S&F Ref: 611320

ORIGINAL

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION FOR THE INVENTION ENTITLED:

A Method of Adaptively Interfacing with a Data Repository

Name and Address of Applicant:

Tenix Industries Pty Limited, an Australian Company, ACN 088 304 652, of Level 1, 100 Arthur Street, North Sydney, New South Wales, 2060, Australia

Names of Inventors:

Michael John Sykes and Daniel Seth Weinstein and Jason Scott Beer

This invention is best described in the following statement:

A METHOD OF ADAPTIVELY INTERFACING WITH A DATA REPOSITORY

FIELD OF THE INVENTION

The present invention relates generally to database systems, and in particular to data warehousing techniques.

BACKGROUND

All types of organisations, business entities, and persons may own legacy database systems that have been acquired at different times. A business may rely upon a particular database or transaction system to handle data aggregation and processing functions for a part of a business. Because of investment, knowledge, and experience with that system, an organisation or entity may choose not to replace such a system, for example simply to avail itself of the stability of the system. Later, another database or transaction system may be acquired and used to handle a different aspect of the business. In this manner, an entity may ultimately operate a number of database systems that do not interact well or at all with one another. In a similar manner, an entity may have its own database or transaction system and need to interact with a number of different database or transaction systems of other entities. For example, a number of entities may be working collaboratively on a project, but each have their own database or transaction systems.

One approach to resolving this problem is to mandate the use of a standard database system throughout the entity or entities. However, this may not be possible or desirable for a number of reasons. For example, an entity working collaboratively with others for a short-term project may consider this to be too onerous of a requirement and therefore unjustifiable.

Data warehouses have attempted to address this problem to collect data from various sources, but suffer from a number of disadvantages. However, such a data warehouse produces mismatches in the data. This results from errors in the data

itself (e.g. due to data entry problems), synchronization problems (e.g., a database may not yet have been amended), and conceptual differences. Relevant conceptual differences include like fields not having the same name, unlike fields having the same name, like fields having different definitions and/or formats, and like entities having different attributes, to name a few.

There has been little synergy between various databases in such circumstances, and users may need to learn a number of different application to find information the users need from such disparate databases.

A further difficulty with such disparate database systems is that each typically has a different interface for the particular database system. Consequently, users must learn a number of different interfaces and become knowledgeable in the nuances of the different interfaces when looking for the same thing across several systems.

Existing user interfaces are written to suit particular datasets. Most often navigation through various screens starts with a menu, from which users select the next screen. This functionality is hardcoded for the particular dataset. Consequently, changes to the data tier require corresponding changes to the application. This means that (1) the user interface is only useful for the original dataset, (2) significant changes to the dataset, or a new dataset, often require a complete rewrite of the user interface, and (3) maintenance and modifications are high cost.

Thus, a need clearly exists for an improved interface technology for accessing data in a data repository that is adaptable to the content of the data repository.

SUMMARY

In accordance with a first aspect of the invention, there is a method of providing an adaptive user interface to a data repository. A data repository is

provided having associated meta-data. The user interface is dynamically generated having interface elements that are dependent upon the meta data. Operation of the interface is controlled by events that also are dependent upon data in the data repository and the meta-data. A browser may be used to access the user interface using a browser, and web pages may be generated for delivery to the browser to provide the user interface.

Preferably, the generating step includes the steps of: checking the data repository to ensure the database has associated meta-data, the meta-data defining relationships in the data repository; and building a menu as a tree object using the meta-data, levels in the tree being built from the meta-data.

Preferably, the interface elements include a main menu, and further including the step of building the main menu dependent upon the meta data, the main menu being an expandable, hierarchically structured object. A search screen, a list screen or a detail page may be invoked if a menu item is selected.

The meta-data may include any one or more of the following: sort order, display name, hierarchy, table ID, target object, navigation URL, and initial expansion.

In accordance with further aspects of the invention, there are an apparatus and a computer program product for providing an adaptive user interface to a data repository, in accordance with the method of the foregoing aspect.

BRIEF DESCRIPTION OF THE DRAWINGS

A small number of embodiments of the invention are described hereinafter with reference to the drawings, in which:

Fig. 1 is a block diagram of system utilising an interface that adaptively accesses and displays data in a data repository in accordance with an embodiment of the invention;

Fig. 2 is a flow diagram illustrating the process of adaptively interfacing with a data repository of Fig. 1;

Figs. 3-12 are screenshots illustrating various aspects of the user interface produced by the process of Fig. 2.

DETAILED DESCRIPTION

A method, a system, and a computer program product for adaptively interfacing with a data repository are described. Numerous specific details are set forth in the following description including data interchange formats, database systems, and the like. However, it will be apparent to those skilled in the art in the light of this disclosure that modifications and/or substitutions may be made without departing from the scope and spirit of the invention. In other instances, well-known details may be omitted so as not to obscure the invention.

Some portions of the description that follows are explicitly or implicitly presented in terms of algorithms and representations of operations on data within a computer system or other device capable of performing computations. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical, or magnetic, or electromagnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that the above and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise, and as apparent from the following, it will be appreciated that throughout the present specification, discussions utilizing terms such as "executing", "selecting", "building", "receiving", "displaying", "storing" "launching", "reporting", or the like, refer to the action and processes of a computer system, or similar electronic device, that manipulates and transforms data represented as physical (electronic) quantities within the registers and memories of the computer system into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present specification also discloses an apparatus or system for performing the operations of the methods. Such an apparatus may be specially constructed for the required purposes, or may include a general-purpose computer or other device selectively activated or reconfigured by a computer program stored in the computer. The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose machines may be used with programs in accordance with the teachings herein. Alternatively, the construction of more specialized apparatus to perform the required method steps may be appropriate. The structure of a conventional general-purpose computer appears from the description below.

In addition, the present invention also implicitly discloses a computer program(s) or software, in that it would be apparent to the person skilled in the art that the individual steps of the preferred method described herein may be put into effect by computer code. The computer program is not intended to be limited to any particular programming language and implementation thereof. It will be appreciated that a variety of programming languages and coding thereof may be used to implement the teachings of the disclosure contained herein. Moreover, the computer program is not intended to be limited to any particular control flow. There are many other variants of the computer program, which can use different control flows without departing the

spirit or scope of the invention. Furthermore one or more of the steps of the computer program may be performed in parallel rather than sequentially.

Such a computer program may be stored on any computer readable medium. The computer readable medium may include storage devices such as magnetic or optical disks, memory chips, or other storage devices suitable for interfacing with a general-purpose computer. The computer readable medium may also include a hard-wired medium such as exemplified in the Internet system, or wireless medium such as exemplified in the GSM mobile telephone system. The computer program when loaded and executed on such a general-purpose computer effectively results in an apparatus that implements the steps of the preferred method.

The preferred method(s) comprise a particular control flow. There are many other variants of the preferred method(s),s which use different control flows without departing the spirit or scope of the invention. Furthermore one or more of the steps of the preferred method(s) may be performed in parallel rather sequential.

Overview

The embodiments of the invention provide a User Interface (labelled Crossbow in Fig. 1) designed to connect to an underlying database and provide the user with powerful search, navigation, display and reporting features. Importantly, the User Interface can be connected to any database that conforms to a defined standard or framework. Having been "pointed" at a new database, the User Interface automatically generates and displays the search and navigation elements, such as Menus, Search Grids and Displays. The User Interface application is driven by the underlying database. The Interface application can be provided details of the database as a parameter(s) of the application to "point" at the database. For web delivery, this may involve providing a URL to a location with a script to run the interface application with details of the database as a parameter.

The User Interface builds these elements by interrogating meta-data in the underlying database, which provides information on where text, data and input objects

are to be displayed. From other metadata tables, the User Interface decides which data is displayed for a particular user and whether the users have read and write access. For example, menus of the interface change when the User Interface is pointed at a new or different database. The User Interface accepts different sets of data and allows a user to navigate and search the data flexibly with minimal or no reprogramming. The User Interface makes decisions about the interface elements on the fly using meta-data tables.

The User Interface utilises intelligent objects invoked by the user while navigating through the interface screens (i.e., event driven). The objects decide when, if and how the objects should react to the events, depending on the data itself, the particular user, and the data stored in the meta-data tables. Because column headings and widths, sort order, colour, fonts, menu items and dropdown box options are all data driven, the User Interface does not care what the data actually represents. The look and feel of the User Interface is driven by Cascading Style Sheets (CSS). The style sheet to use can also be data driven.

Before proceeding further with a more detailed description of the User Interface process, a brief overview is provided of a representative system with which the embodiments of the invention may be practiced.

System 100

Fig. 1 is a block diagram illustrating an embodiment of the invention for adaptively interfacing with a data repository 130. The data repository 130 may be any data set that satisfies a specified framework for the User Interface. The main elements of system 100 include a data repository 130 (e.g., a data warehouse) and a User Interface system 170 (labelled Crossbow) coupled to the data repository 130 via a web server 160. The User Interface system 170 is preferably implemented as an ASP.Net application, which allows the application to be browser-based. Application Service Provider (ASP) is a specification for dynamically created Web pages with a .ASP extension that utilizes ActiveX scripting (e.g., Visual Basic Script or Java script

code). When a browser requests an ASP page, the web server 160 generates a page with HTML code and sends the page back to the requesting browser. The User Interface is preferably implemented using Library Objects in the form of reuseable dll files. Remote users using client systems 190, 192 are coupled to the web server 160 via a network 180, such as the Internet or an Intranet. The remote users can access the data repository via the User Interface 170 using a web browser. The system 100 is therefore capable of delivering data to any browser user in the world that has Internet access.

Preferably but not necessarily, the data repository 130 is derived from several different database or transaction systems 110. This includes a number of individual transaction systems 110A, 110B, ..., 110C, which periodically load data 112 into the data warehouse 130. The individual transaction systems 110A, 110B, ..., 110C may poorly interact with one or more of the other transaction systems, or not at all. Preferably but not necessarily, a staging area 122 receives the data 212 periodically loaded from the transaction systems 210. The staging area 122 receives both good and bad data. Data transform rules are applied between the transaction systems and the staging area, which may produce an intermediate file. Data may be brought into the staging area 112 using variable character field text, for example. Preferably but not necessarily, a virtual quality firewall 120 is maintained between the staging area 122 and the data warehouse 130 to keep bad data out of the data repository 130 and to identify errors in the data from the staging area 122. The data warehouse 130 includes warehouse data 132 and meta-data 134. Preferably but not necessarily, the data warehouse 130 also includes a data history 136, an error log 138, an error history 140, and stored procedures 142.

Main Menu

The data necessary to build a Main Menu using the User Interface 170 is retrieved from the meta data 134 and is used to populate an expandable hierarchal or "Tree" structured object that is the Main Menu. The meta-data 134 provides information such as the following:

- sort order,
- display name,
- hierarchy,
- table ID,
- target object,
- navigation URL, and
- initial expansion.

Selecting a Menu item invokes a Search Screen, List Screen, or a Detail Page, described hereinafter.

Search Screen

Each column of every table is defined in a Columns Table as part of the meta-data 134. Attributes of each column define the column's participation in objects such as Search Screens. This data is used to generate each Search Screen.

List Screen

List screens are driven by a custom built List Object. From using the meta-data 134, the list object generates lists of rows returned by the Search Screen SQL "Where" clause. Column participation, headings and hyperlinks are all generated from the meta-data. Selection of a row by the user invokes the appropriate Detail Page.

Detail Page

Detail Pages are related to tables (and hence entities in the data). Table and Column related meta-data 134 dictates participation of that column, its position on the screen, user access and the related text heading.

Related Data

Every Detail Page also has a "Related Data" drop-down box. Population of this box comes from a meta-data table that describes all table joins. Depending on

the type of the relationship (parent, child or sibling), the word "All" or "The" is added to the beginning of the display name. For children, where "All" is added to the beginning, an "s" is added to pluralise the last word. For example, consider the situation where a Detail Page displayed the entity "Customer" and a relationship is described in the Join table where a child table contained many Orders relating to that Company. In this case, the display name for the Orders and Customer tables are "Order" and "Customer" respectively. The listing in the drop-down box is "All Orders", and when an Order is the subject of a Detail Page, the drop-down box entry reads "The Customer".

Reporting

Because all Search Screens accept Wild Cards, the Search Screens can return almost any set of rows as a subset of the data. These rows are used to generate either Spreadsheet or database files using Microsoft Access™ or Excel™ files. Users can then either use the data directly, or manipulate the data further as necessary.

Managing Meta Data

Because the User Interface 170 can also access meta-data tables, the meta-data 134 can be treated just like any other data in the database 130. Therefore, most of the meta-data 134 can be edited in a series of Administration Screens (see Fig. 12). The relationship between the meta-data tables is also described in the Join, Column and Table tables. Therefore, the meta-data tables can also be navigated in the same way that other data is treated. These and other aspects are described in greater detail hereinafter with reference to Fig. 2.

User Interface Process 200

Fig. 2 is a flow diagram illustrating the process 200 of adaptively interfacing with a data repository derived from the different transaction systems of Fig. 1. Processing commences in step 202. In step 204, a main menu is built using meta-data 206 (134 in Fig. 1), as described hereinbefore. From step 204, two events 210 and 212 may occur. In step 210, an event can occur in which the user selects a

menu item. A menu item provides an entity name. In parallel with step 210, processing from step 204 can continue at step 212. In step 212, an event can occur in which the user selects a hierarchical view (see Fig. 11). There may be two or more hierarchical views. A hierarchical view is an alternate way of getting at data. In step 214, the hierarchy is displayed dependent upon meta-data 206 and a cached hierarchy 208. Because the hierarchy does not change between data loads, the hierarchical structure may be cached. The processing and resources required to build the hierarchical view of the data is consequently only done once per data load in such circumstances. For some entities in the data, there is a relationship between rows in a table, i.e. a recursive relationship. Within a table, the relationship may be displayed as a hierarchy. Normally, a table is searched and a parent-child relationship is built between rows. The cached hierarchy stores the parent information about a row. From step 214, an event can occur step 216 in which the user selects a hierarchical item from the hierarchical display. From step 216, processing continues at step 248. From step 214, processing can return to step 210. From step 210, processing continues at step 218.

In decision step 218, a check is made to determine if a search screen is required. A search screen is built from a menu item. If step 218 returns false (NO), processing continues at step 236. In step 236, a Related List screen is displayed. Otherwise, if step 218 returns true (YES), processing continues at step 220. In step 220, a Related List Search screen is displayed. In step 222, an event can occur in which a user enters search criteria and selects a "Find" command. Search criteria may be entered in one or more fields, and the user may specify wild cards. In step 224, a SQL "Where" clause is built based on the search criteria from step 222. In step 226, a "List" object is called. The list object contains data returned from the "where" clause. The list object 228 involves steps 230, 232, and 234. In decision step 230, a check is made to determine if the SQL command returns more than one row. If step 230 returns false (NO), processing continues at step 248. If zero rows are returned, a message is displayed stating that no records have been found (not shown). Otherwise, if step 230 returns true (YES), processing continues at step 232. In decision step 232,

a check is made to determine if the row count exceeds a given or predetermined number (e.g., 500). If decision step 232 returns false (NO), processing continues at step 236. Otherwise, if step 232 returns true (YES), processing continues at step 234. In step 234, the display rows are limited to the predetermined or given number of step 232 (e.g., 500), and all rows are cached. Processing then continues at step 236.

In step 236, a Related List screen is displayed dependent upon meta-data 238. From step 236, three events 238, 244, and 246 may occur. In step 239, an event can occur in which the user selects a reporting option, following step 236. In step 240, a report is built from the list or cached rows. In step 242, a database or spreadsheet application (e.g., Microsoft Access or Microsoft Excel) is launched and populated with row data. Otherwise, from step 236, processing may continue at step 244 or 246. In step 244, an event occurs in which a user selects a list item. In step 246, the user selects a hyperlink. A column heading may be underlined allowing the user to move from the location to an item. From steps 244 or 246, processing continues at step 248.

In step 248, a Related Detail screen is displayed based on meta-data 256 and screen layout information 254. In step 250, a Related Information Drop Down box is built dependent upon meta-data 256. This step uses a join table. Meta data is interrogated to find all possible relationships to other entities in the database. From step 250 an event can occur in step 252 in which the user selects the drop down box item. From step 252, four events 210, 258, 260, and 262 can occur. In step 258, an event can occur in which the user selects a "Parent" item. In step 260, an event can occur in which the user selects a "Child" item. In step 262, an event can occur in which the user selects a "Sibling" item. From each of steps 258, 260, and 262, processing continues at step 226.

Thus, by using meta-data 134 to describe the relationship between the various elements, a single User Interface can be used to search, navigate, display, edit and report data from many different databases. Simply designing a new data tier can

generate new end-user applications. The User Interface is also a software application development environment. Developers need only develop their application at the data tier level. The User Interface then becomes the user interface for the application, automatically adapting to the new data. This saves design effort and the necessity to program a new User Interface. The adaptability of the User Interface is achieved by using meta-data tables within the data repository. Such tables are easier to generate and maintain than rewriting the software application. The User Interface allows a user to start from any point, with whatever piece of data the user has, and then search and navigate through the data repository to the needed information. The User Interface can easily be transported to other sets of data. Adding additional data is relatively simple.

The User Interface may also be implemented as a standalone application together with its own data. The data may be periodically updated using a CDROM, or via network access, for example.

Screenshots

Several representative screenshots are depicted in Figs. 3-12 to illustrate the appearance of the interface produced by the process 200 of Fig. 2. The screenshots are of a browser utilising the adaptive user interface to access data. In the screenshots, the Microsoft Internet Explorer browser is used.

Fig. 3 illustrates a screenshot 300 of the interface produced. The main menu 310 is shown for the current database or repository pointed to. A number of items are shown in the main menu 310, including an item 320 "As-built Configuration Part". Also shown in the interface is a time frame, indicating the stored history. If the item 320 is clicked on, the screenshot 400 of the interface is produced, labelled "As-built Configuration Part". The search screen 410 is shown. The fields shown in the search screen 410 and what type of fields is dependent upon the meta-data. As shown in Fig. 5, by filling in the Description field 520 in the search screen 510 with "diesel" and clicking find, the list grid 600 of 175 items is produced. Each of the items may be

clicked on for more details. Clicking on the topmost item 610 produces a related detail screen (248 in Fig. 2).

Fig. 7 is the resulting screenshot 700 of the related detail screen, from clicking on item 610. This is the result of a "Where" statement built using "diesel" from the search screen. A related information box 720 is depicted with "None" specified. Fig. 8 is a screenshot 800 that results by selecting "All Serialised Documents" in the box 720 of Fig. 7. One serialised document 820 is depicted as a result. This is a related item "child" (260 in Fig. 2).

Fig. 9 is a screenshot 900 depicting details of a SQL query 910, produced by step 224 of Fig. 2. This can be shown by clicking on the SQL icon in Fig. 9.

Fig. 10 is a screenshot 1000 providing further details of the main menu. One of the items in the main menu is the Administration item, which has a child element 1010 "Error Statistics by Table". This is an example of an administrative function available through the User Interface. A list 1020 of 51 tables and the corresponding error count is produced.

Fig. 11 is a screenshot 1100 showing a hierarchical view (step 214 of Fig. 2) showing all links 1110 within the tree. The links 1110 are hyperlinks to items in the database. Each represents a row.

Finally, Fig. 12 is a screenshot 1200 illustrating meta-data administrative functions 1210 illustrating that the interface can be used to access and modify the meta data tables.

Thus, the embodiments of the invention provide a largely generic user interface, capable of attaching to a wide set of databases. The initial menu and all sub-menus are all generated from the underlying database. Most of the necessary

information is held in pre-defined meta data tables. The advantages of this approach include:

- A single user interface can be reused in a number of applications;
- New applications can be written quickly and cheaply;
- Users skilled in one application can learn a new application more quickly;
- Changes to the data tier often do not require changes to the user interface; and
- In summary, applications are easier to develop and maintain, and therefore are less expensive.

Computer Implementation

The method of adaptively interfacing with a data repository is preferably practiced using one or more general-purpose computer systems, in which the processes of Figs. 1 and 2 may be implemented as software, such as an application program executing within the computer system or handheld device. In particular, the steps of method of adaptively interfacing with a data repository are effected, at least in part, by instructions in the software that are carried out by the computer. The instructions may be formed as one or more code modules, each for performing one or more particular tasks. The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer program product.

Examples of computers on which the described arrangements can be practised include IBM-PC's and compatibles, Sun Sparcstations or alike computer systems. Still further, the software can also be loaded into the computer system from other computer readable media. The term "computer readable medium" as used herein refers to any storage or transmission medium that participates in providing instructions and/or data to the computer system for execution and/or processing. Examples of storage media include floppy disks, magnetic tape, CD-ROM, a hard

disk drive, a ROM or integrated circuit, a magneto-optical disk, or a computer readable card such as a PCMCIA card and the like, whether or not such devices are internal or external of the computer module. Examples of transmission media include radio or infra-red transmission channels as well as a network connection to another computer or networked device, and the Internet or Intranets including e-mail transmissions and information recorded on Websites and the like.

A small number of embodiments of the invention regarding a method, a system, and a computer program product for adaptively interfacing with a data repository have been described. In the light of the foregoing, it will be apparent to those skilled in the art in the light of this disclosure that various modifications and/or substitutions may be made without departing from the scope and spirit of the invention.

The claims defining the invention are as follows:

1. A method of providing an adaptive user interface to a data repository, said method including the steps of:

providing a data repository, said data repository having associated meta-data; and

dynamically generating said user interface having interface elements that are dependent upon said meta data, operation of said interface controlled by events that also are dependent upon data in said data repository and said meta-data.

2. The method according to claim 1, wherein said generating step includes the steps of:

checking said data repository to ensure said database has associated meta-data, said meta-data defining relationships in said data repository; and

building a menu as a tree object using said meta-data, levels in said tree being built from said meta-data.

3. The method according to claim 1, further including the steps of:

accessing said user interface using a browser; and

generating web pages for delivery to said browser to provide said user interface.

4. The method according to claim 1, wherein said interface elements include a main menu, and further including the step of building said main menu dependent upon said meta data, said main menu being an expandable, hierarchically structured object.

5. The method according to any one of claims 1 to 4, wherein said meta-data includes any one or more of the following:

sort order,
display name,
hierarchy,
table ID,
target object,
navigation URL, and
initial expansion.

6. The method according to claim 4, further including the step of invoking a search screen, a list screen or a detail page if a menu item is selected.

7. An apparatus for providing an adaptive user interface to a data repository, said apparatus including:

a data repository having associated meta-data; and

means for dynamically generating said user interface having interface elements that are dependent upon said meta data, operation of said interface controlled by events that also are dependent upon data in said data repository and said meta-data.

8. The apparatus according to claim 7, wherein said generating means includes:

means for checking said data repository to ensure said database has associated meta-data, said meta-data defining relationships in said data repository; and

means for building a menu as a tree object using said meta-data, levels in said tree being built from said meta-data.

9. The apparatus according to claim 7, further including:

means for accessing said user interface using a browser; and

means for generating web pages for delivery to said browser to provide said user interface.

10. The apparatus according to claim 7, wherein said interface elements include a main menu, and further including means for building said main menu dependent upon said meta data, said main menu being an expandable, hierarchically structured object.

11. The apparatus according to any one of claims 7 to 10, wherein said meta-data includes any one or more of the following:

sort order,
display name,
hierarchy,
table ID,
target object,
navigation URL, and
initial expansion.

12. The apparatus according to claim 10, further including means for invoking a search screen, a list screen or a detail page if a menu item is selected.

13. A computer program product for providing an adaptive user interface to a data repository, said computer program product including:

computer program code means for providing a data repository having associated meta-data; and

computer program code means for dynamically generating said user interface having interface elements that are dependent upon said meta data, operation of said interface controlled by events that also are dependent upon data in said data repository and said meta-data.

14. The computer program product according to claim 13, wherein said computer program code means for generating includes:

computer program code means for checking said data repository to ensure said database has associated meta-data, said meta-data defining relationships in said

data repository; and

computer program code means for building a menu as a tree object using said meta-data, levels in said tree being built from said meta-data.

15. The computer program product according to claim 13, further including:

computer program code means for accessing said user interface using a browser; and

computer program code means for generating web pages for delivery to said browser to provide said user interface.

16. The computer program product according to claim 13, wherein said interface elements include a main menu, and further including computer program code means for building said main menu dependent upon said meta data, said main menu being an expandable, hierarchically structured object.

17. The computer program product according to any one of claims 13 to 16, wherein said meta-data includes any one or more of the following:

sort order,
display name,
hierarchy,
table ID,
target object,
navigation URL, and
initial expansion.

18. The computer program product according to claim 16, further including computer program code means for invoking a search screen, a list screen or a detail page if a menu item is selected.

DATED THIS FOURTH DAY OF OCTOBER, 2002

TENIX INDUSTRIES PTY LIMITED

PATENT ATTORNEYS FOR THE APPLICANT

SPRUSON & FERGUSON

A METHOD OF ADAPTIVELY INTERFACING WITH A DATA REPOSITORY

ABSTRACT

A method (200), a system (100), and a computer program product for adaptively interfacing with a data repository (130) are disclosed. A data repository (130) having associated meta-data (134) is provided. The user interface (170) is dynamically generated having interface elements that are dependent upon the meta-data (134). Operation of the interface (170) is controlled by events (210, 212, 216, 222, 238, 244, 246, 252, 258, 260, 262) that also are dependent upon data in the data repository (130) and the meta-data (134). The user interface (170) may be accessed using a browser (190, 192), and web pages may be generated for delivery to the browser (190, 192) to provide the user interface (170).

Fig. 1

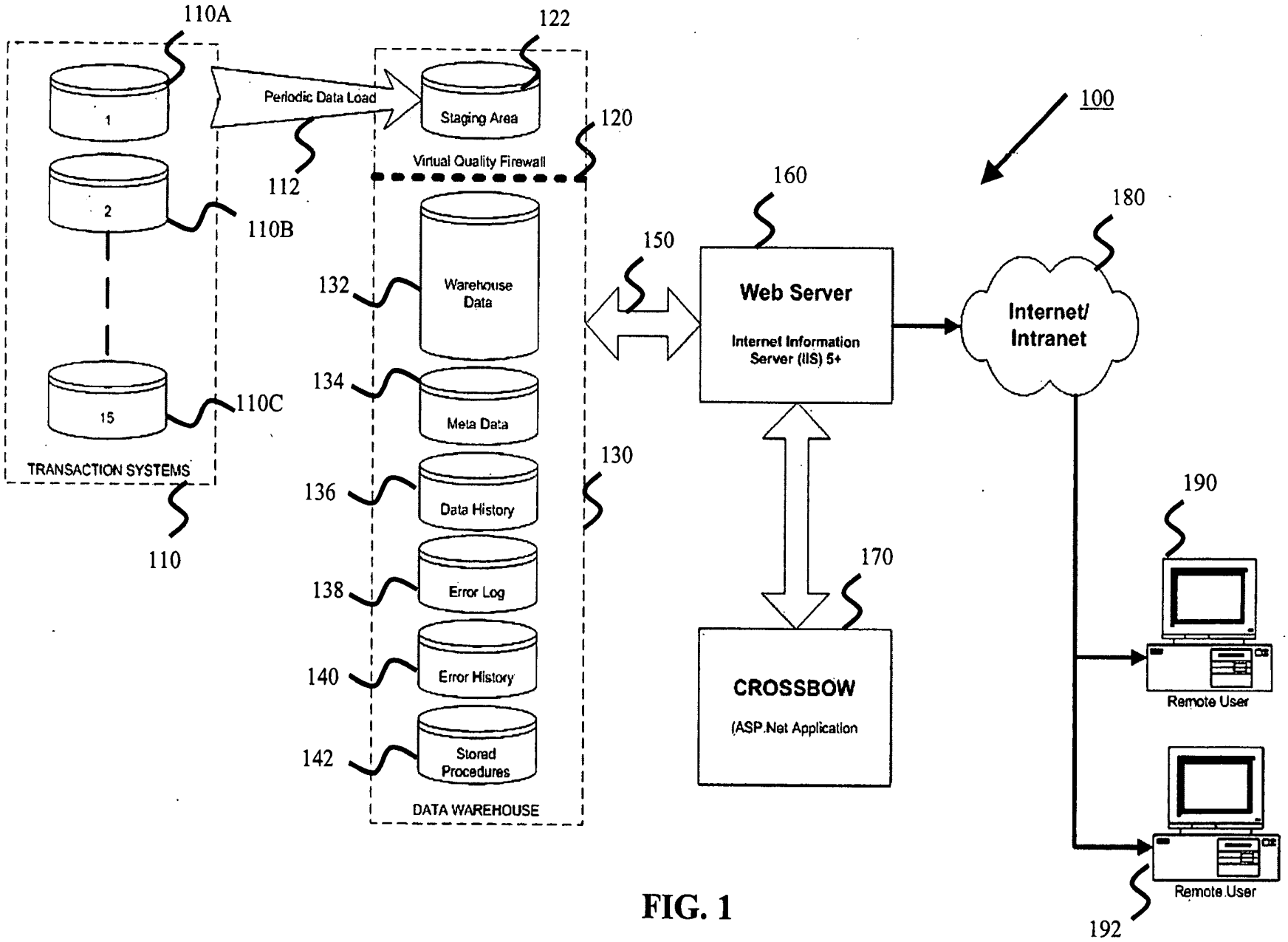
110A
122
1
Periodic Data Load
Staging Area
120
Virtual Quality Firewall
112
2
110B
100
160
180
150
132
Warehouse Data
Web Server
Internet Information Server (IIS) 5+
Internet/ Intranet
15
110C
134
Meta Data
130
TRANSACTION SYSTEMS
136
Data History
190
110
138
Error Log
170
140
Error History
CROSSBOW
(ASP.Net Application
Remote User
142
Stored Procedures
DATA WAREHOUSE
Remote User
192

FIG. 1

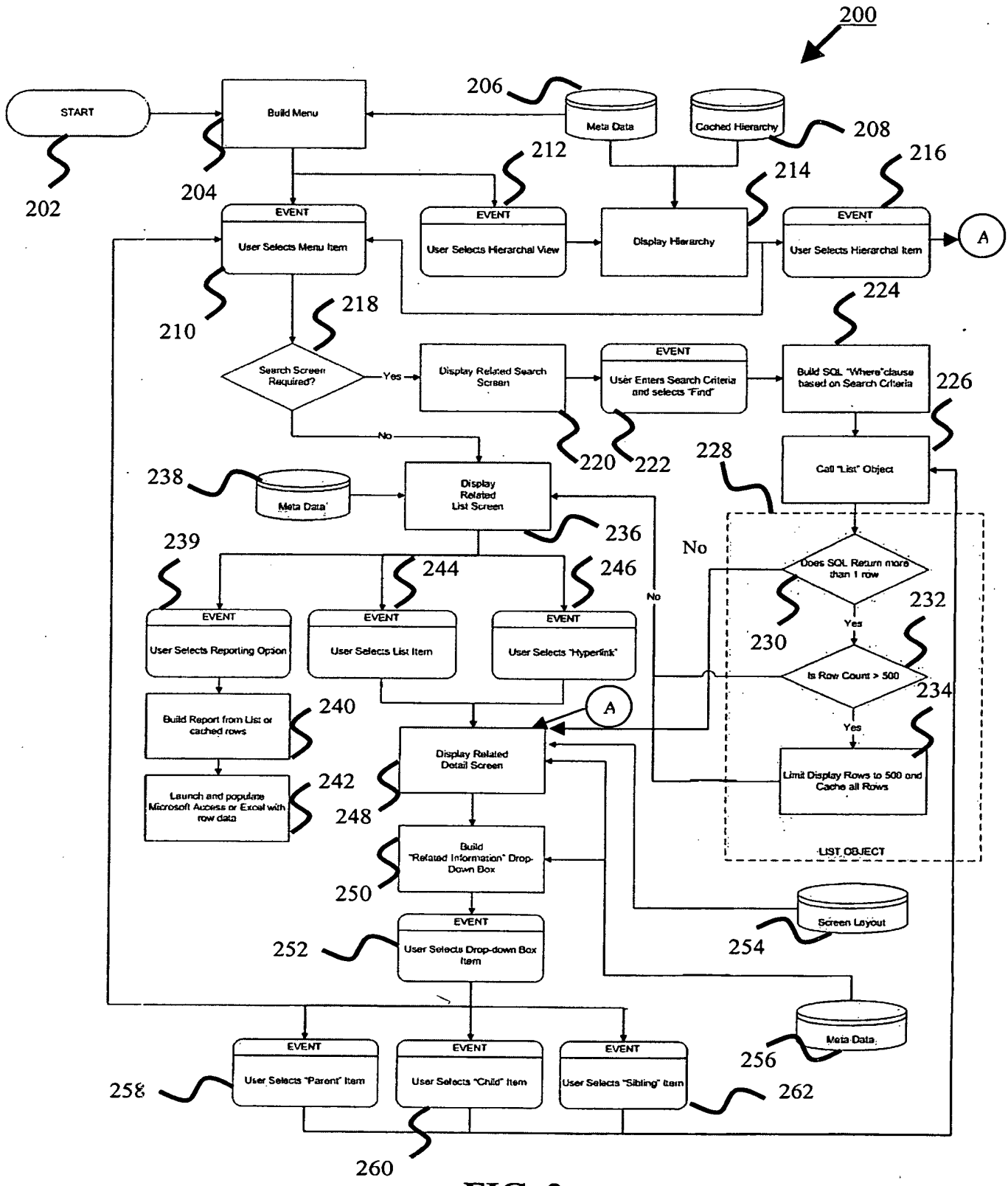
200
START
202
Build Menu
204
206
Meta Data
Cached Hierarchy
208
212
214
216
EVENT
User Selects Menu Item
EVENT
User Selects Hierarchal View
Display Hierarchy
EVENT
User Selects Hierarchal Item
A
210
218
224
Search Screen Required?
Yes
No
Display Related Search Screen
EVENT
User Enters Search Criteria and selects "Find"
Build SQL "Where" clause based on Search Criteria
226
220
222
228
Call "List" Object
238
Meta Data
Display Related List Screen
236
239
244
246
No
Does SQL Return more than 1 row
230
232
Yes
Is Row Count > 500
234
Yes
EVENT
User Selects Reporting Option
EVENT
User Selects List Item
EVENT
User Selects "Hyperlink"
Build Report from List or cached rows
240
A
Limit Display Rows to 500 and Cache all Rows
Display Related Detail Screen
242
248
Launch and populate Microsoft Access or Excel with row data
LIST OBJECT
Build "Related Information" Drop-Down Box
250
Screen Layout
254
EVENT
User Selects Drop-down Box Item
252
Meta Data
256
EVENT
User Selects "Parent" Item
EVENT
User Selects "Child" Item
EVENT
User Selects "Sibling" Item
258
262
260

FIG. 2

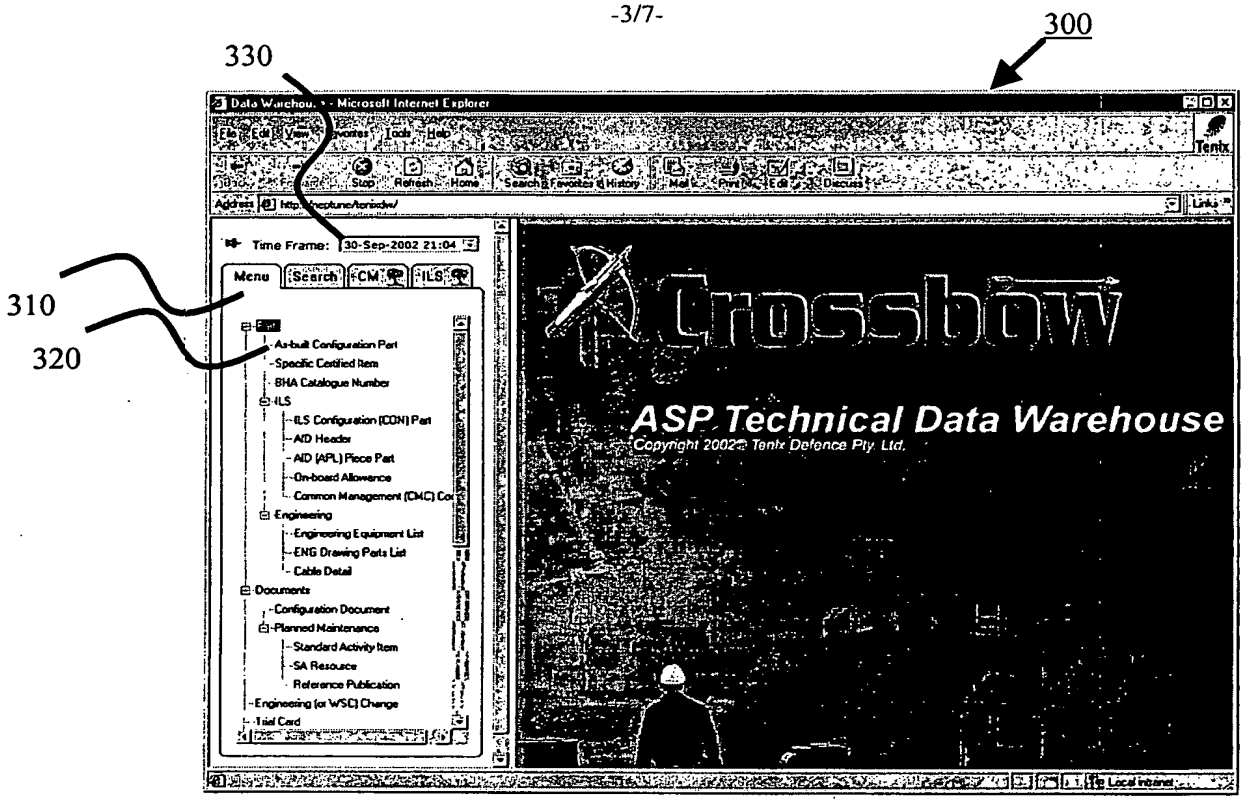
300
330
310
320
Time Frame:
30-Sep-2002 21:04
Menu
Search
CM
ILS
As-built Configuration Part
Specific Certified Item
BHA Catalogue Number
ILS
ILS Configuration (CDN) Part
AID Header
AID (APL) Piece Part
On-board Allowance
Engineering
Engineering Equipment List
ENG Drawing Parts List
Cable Detail
Documents
Configuration Document
Planned Maintenance
Standard Activity Item
SA Resource
Reference Publication
Engineering (or WSC) Change
Crossbow
ASP Technical Data Warehouse
Copyright 2002© Tenix Defence Pty. Ltd.

FIG. 3

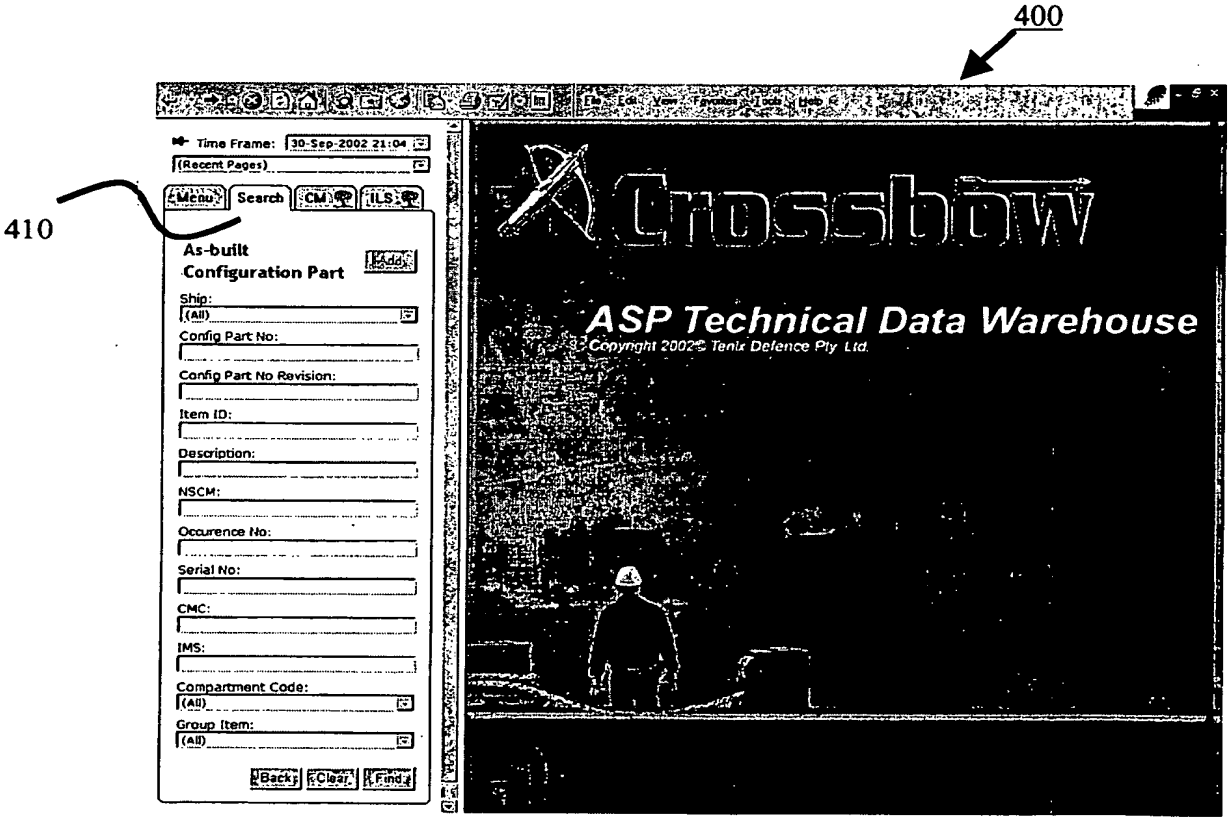
400
410
Time Frame:
30-Sep-2002 21:04
(Recent Pages)
Menu
Search
CM
ILS
As-built Configuration Part
Add
Ship:
(All)
Config Part No:
Config Part No Revision:
Item ID:
Description:
NSCM:
Occurence No:
Serial No:
CMC:
IMS:
Compartment Code:
(All)
Group Item:
(All)
Back
Clear
Find
Crossbow
ASP Technical Data Warehouse
Copyright 2002© Tenix Defence Pty. Ltd.

FIG. 4

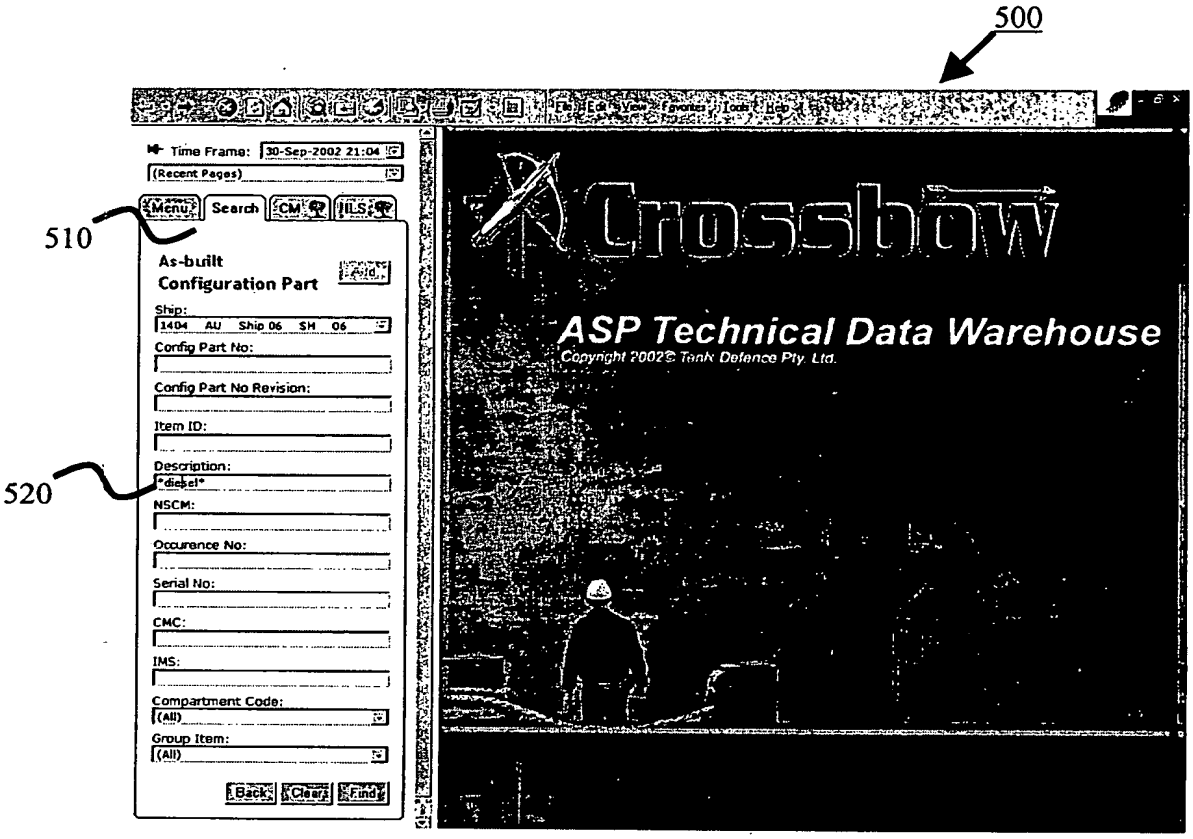
500
510
520
Time Frame: 30-Sep-2002 21:04
(Recent Pages)
Menu
Search
CM
ILS
As-built Configuration Part
Add
Ship:
1404 AU Ship 06 SH 06
Config Part No:
Config Part No Revision:
Item ID:
Description:
diesel
NSCM:
Occurence No:
Serial No:
CMC:
IMS:
Compartment Code:
(All)
Group Item:
(All)
Back
Clear
Find
Crossbow
ASP Technical Data Warehouse
Copyright 2002© Tenix Defence Pty. Ltd.

FIG. 5

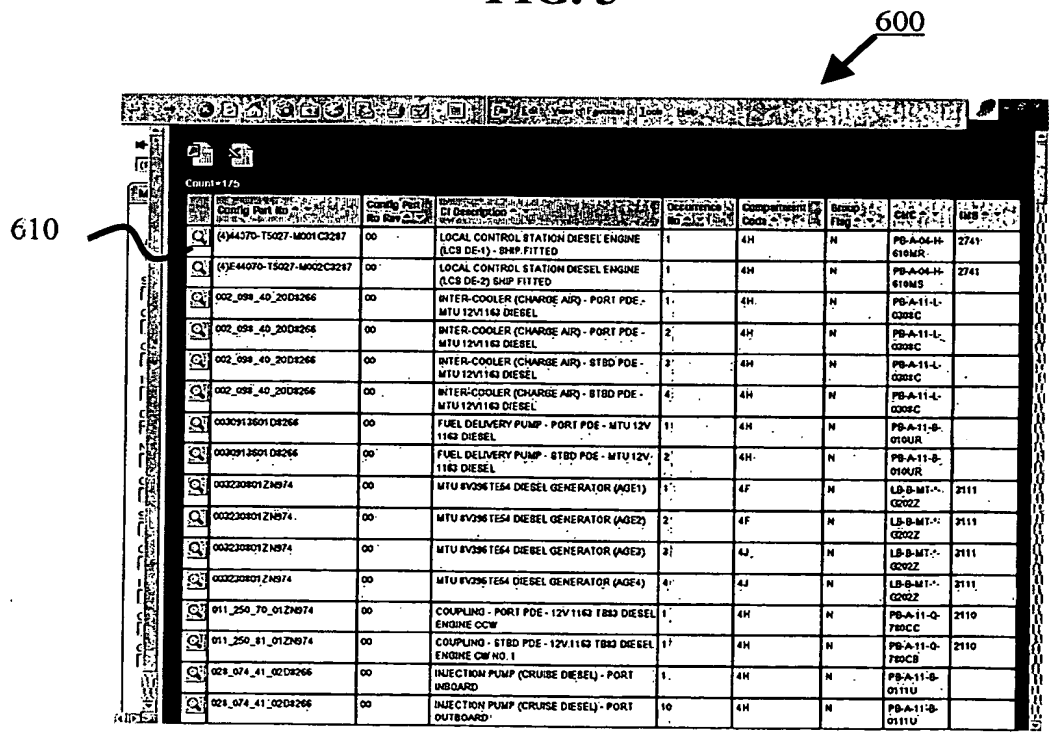
600
610
Count=175
Config Part No
Config Part No Rev
CI Description
Occurrence No
Compartment Code
Group Item
CMC
IMS
(4)44070-T5027-M001C3287
LOCAL CONTROL STATION DIESEL ENGINE (LCS DE-1) - SHIP FITTED
(4)E44070-T5027-M002C3287
LOCAL CONTROL STATION DIESEL ENGINE (LCS DE-2) SHIP FITTED
002_098_40_20D8266
INTER-COOLER (CHARGE AIR) - PORT PDE - MTU 12V1163 DIESEL
INTER-COOLER (CHARGE AIR) - STBD PDE - MTU 12V1163 DIESEL
0030913601D8266
FUEL DELIVERY PUMP - PORT PDE - MTU 12V 1163 DIESEL
FUEL DELIVERY PUMP - STBD PDE - MTU 12V 1163 DIESEL
0032308012N974
MTU 8V396TE54 DIESEL GENERATOR (AGE1)
MTU 8V396TE54 DIESEL GENERATOR (AGE2)
MTU 8V396TE54 DIESEL GENERATOR (AGE3)
MTU 8V396TE54 DIESEL GENERATOR (AGE4)
011_250_70_01ZN974
COUPLING - PORT PDE - 12V1163 TB83 DIESEL ENGINE CCW
011_250_81_01ZN974
COUPLING - STBD PDE - 12V1163 TB83 DIESEL ENGINE CW NO. 1
023_074_41_02D8266
INJECTION PUMP (CRUISE DIESEL) - PORT INBOARD
INJECTION PUMP (CRUISE DIESEL) - PORT OUTBOARD

FIG. 6

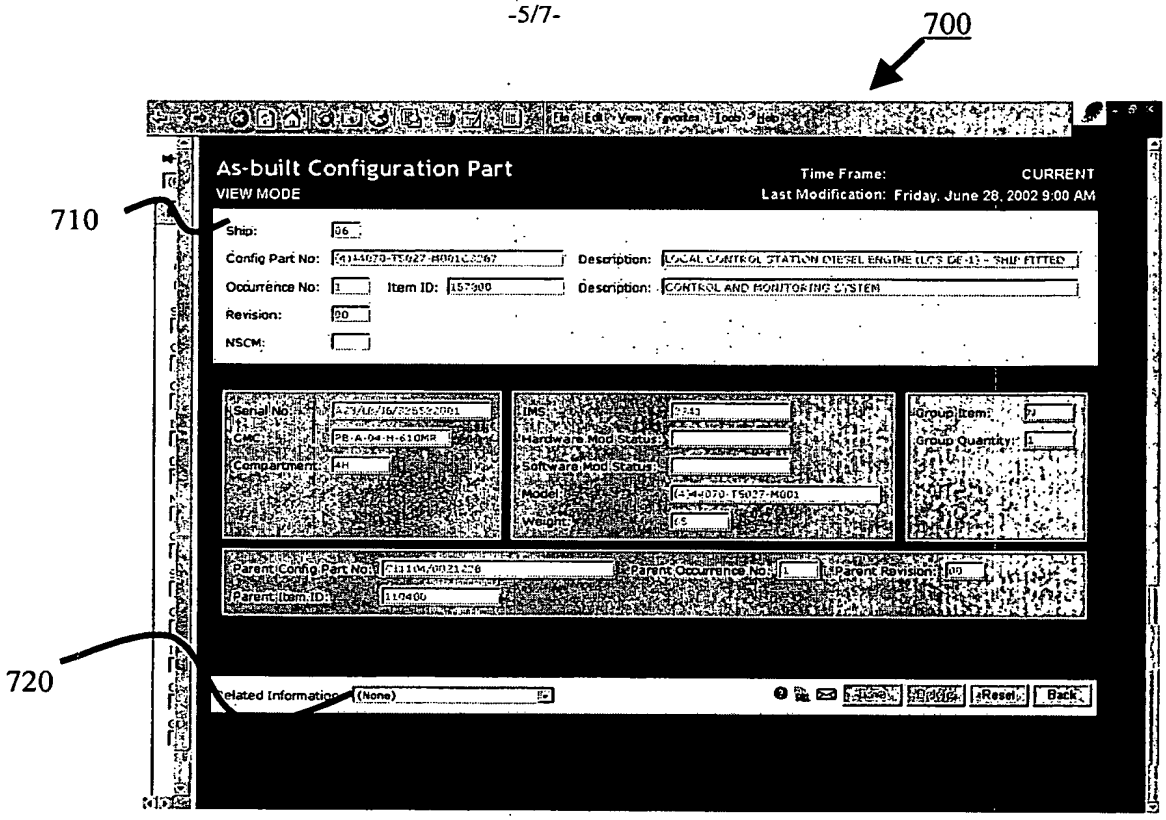
700
710
720
As-built Configuration Part
VIEW MODE
Time Frame: CURRENT
Last Modification: Friday, June 28, 2002 9:00 AM
Ship:
Config Part No:
Description:
Occurrence No:
Item ID:
Revision:
NSCM:
Serial No:
CMC:
Compartment:
Hardware Mod Status:
Software Mod Status:
Model:
Weight:
Group Item:
Group Quantity:
Parent Config Part No:
Parent Occurrence No:
Parent Revision:
Parent Item ID:
Related Information:
(None)
Reset
Back

FIG. 7

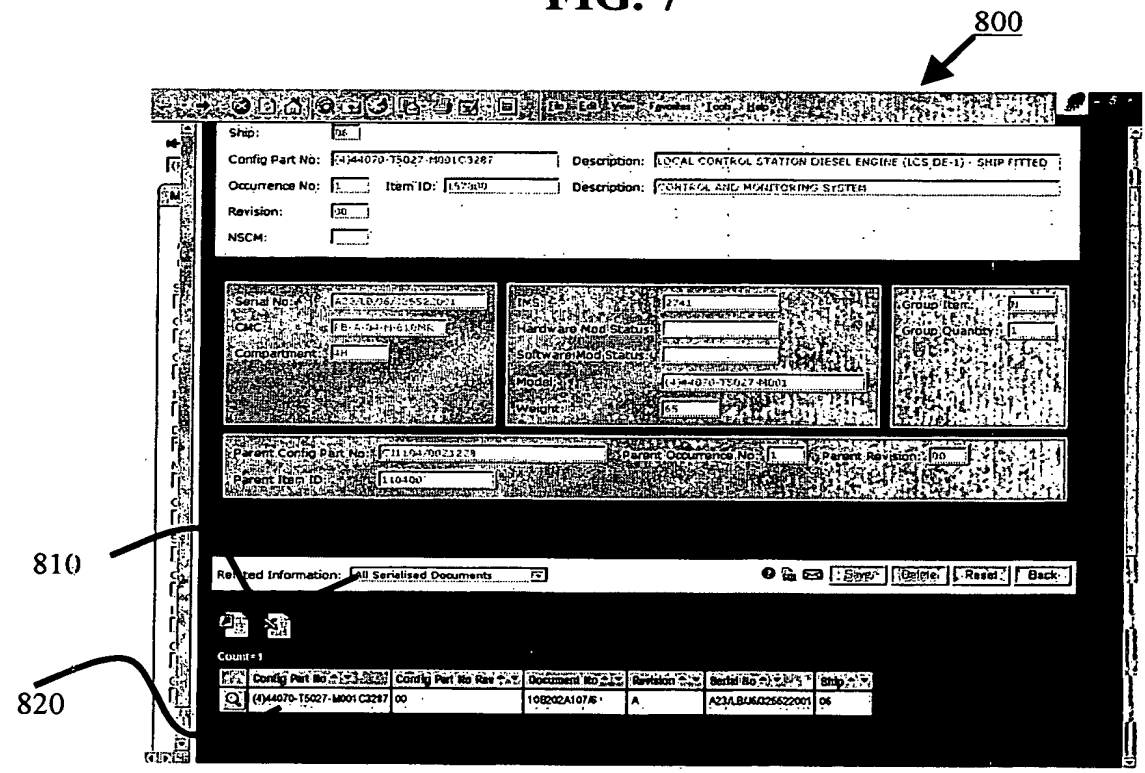
800
810
820
Ship:
Config Part No:
Description:
Occurrence No:
Item ID:
Revision:
NSCM:
Related Information:
All Serialised Documents
Save
Delete
Reset
Back
Count = 1
Config Part No
Config Part No Rev
Document No
Revision
Serial No
Ship
(4)44070-T5027-M001C3287
00
108202A107/6
A
A23/LBJ/6025622001
06

FIG. 8

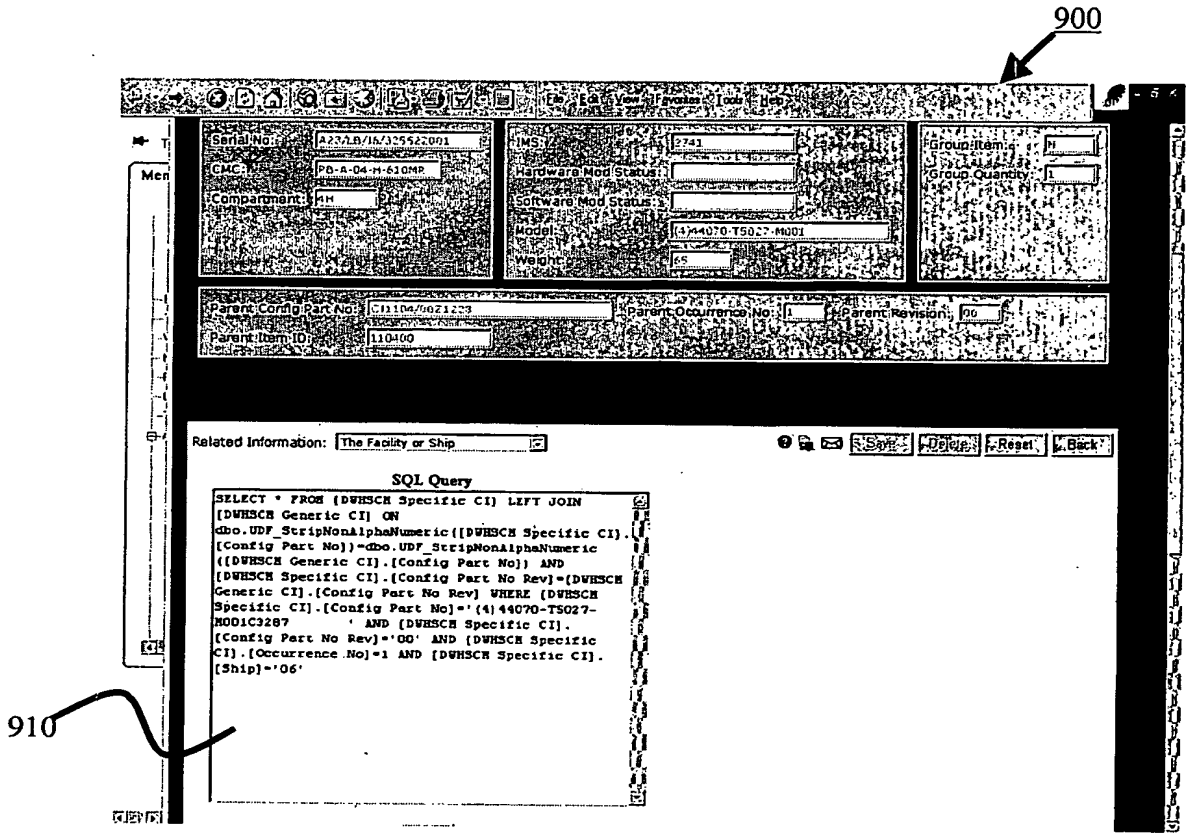
900
Related Information:
The Facility or Ship
SQL Query
910

FIG. 9

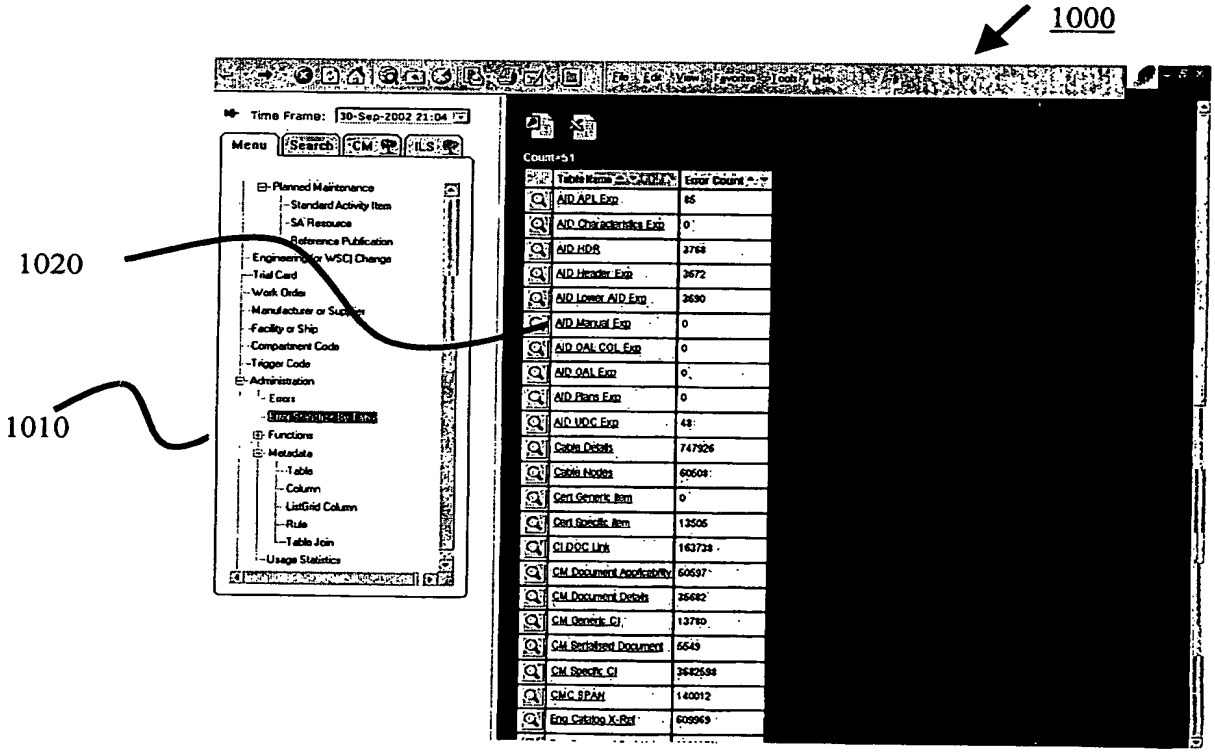
1000
1020
1010
Time Frame:
30-Sep-2002 21:04
Menu
Search
Count=51

FIG. 10

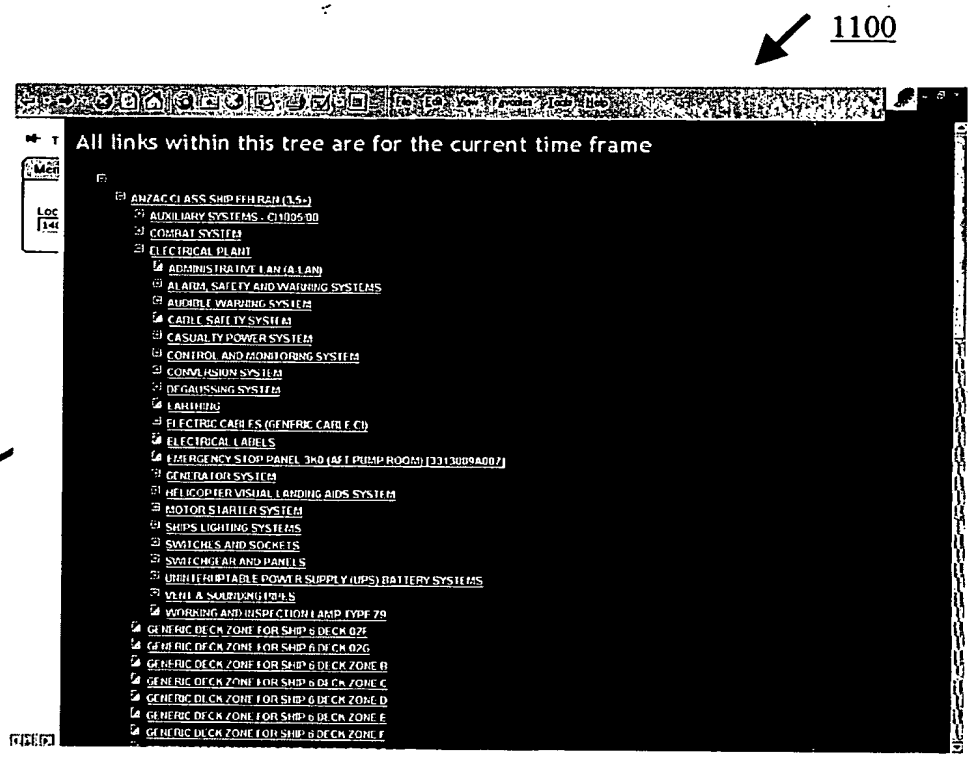
1100
1110
All links within this tree are for the current time frame
ANZAC CLASS SHIP FFH RAN (3.5+)
AUXILIARY SYSTEMS - C1005/00
COMBAT SYSTEM
ELECTRICAL PLANT
ADMINISTRATIVE LAN (A LAN)
ALARM, SAFETY AND WARNING SYSTEMS
AUDIBLE WARNING SYSTEM
CABLE SAFETY SYSTEM
CASUALTY POWER SYSTEM
CONTROL AND MONITORING SYSTEM
CONVERSION SYSTEM
DEGAUSSING SYSTEM
EARTHING
ELECTRIC CABLES (GENERIC CABLE CI)
ELECTRICAL LABELS
EMERGENCY STOP PANEL 3KD (AFT PUMP ROOM) [3313009A007]
GENERATOR SYSTEM
HELICOPTER VISUAL LANDING AIDS SYSTEM
MOTOR STARTER SYSTEM
SHIPS LIGHTING SYSTEMS
SWITCHES AND SOCKETS
SWITCHGEAR AND PANELS
UNINTERRUPTABLE POWER SUPPLY (UPS) BATTERY SYSTEMS
WELL & SOUNDING PIPES
WORKING AND INSPECTION LAMP TYPE 79
GENERIC DECK ZONE FOR SHIP 6 DECK 02E
GENERIC DECK ZONE FOR SHIP 6 DECK 02G
GENERIC DECK ZONE FOR SHIP 6 DECK ZONE B
GENERIC DECK ZONE FOR SHIP 6 DECK ZONE C
GENERIC DECK ZONE FOR SHIP 6 DECK ZONE D
GENERIC DECK ZONE FOR SHIP 6 DECK ZONE E
GENERIC DECK ZONE FOR SHIP 6 DECK ZONE F

FIG. 11

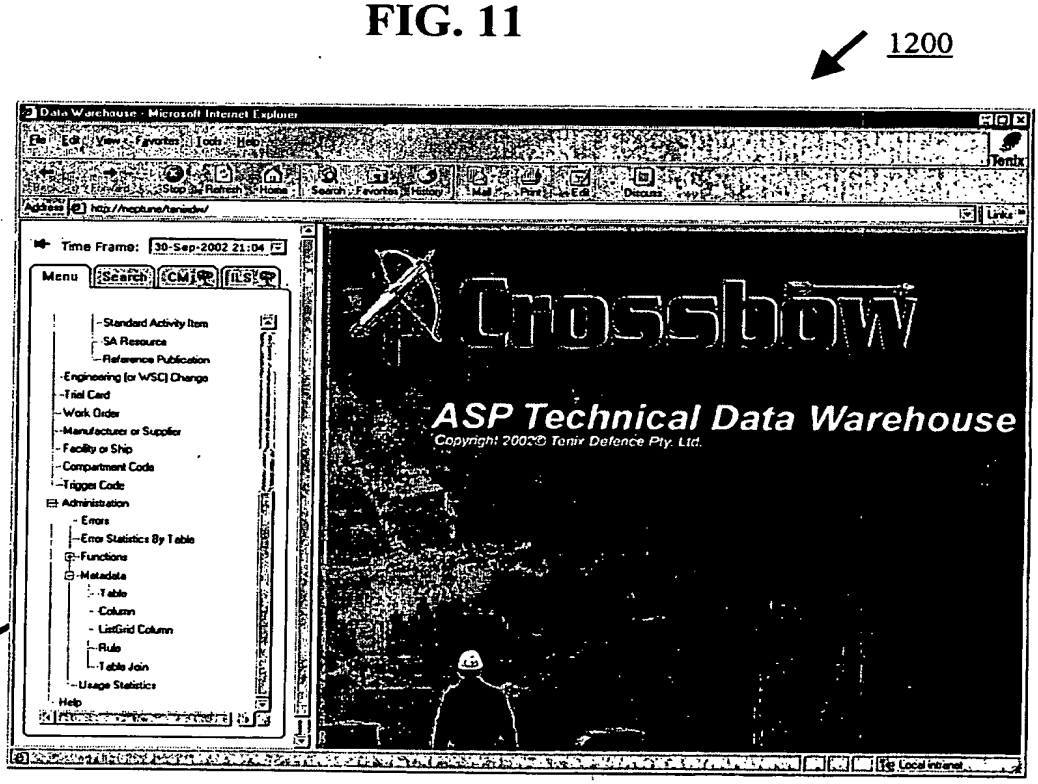
1200
1210
Data Warehouse - Microsoft Internet Explorer
Time Frame:
30-Sep-2002 21:04
Menu
Search
Standard Activity Item
SA Resource
Reference Publication
Engineering (or WSC) Change
Trial Card
Work Order
Manufacturer or Supplier
Facility or Ship
Compartment Code
Trigger Code
Administration
Errors
Error Statistics By Table
Functions
Metadata
Table
Column
ListGrid Column
Rule
Table Join
Usage Statistics
Help
Crosshow
ASP Technical Data Warehouse
Copyright 2002© Tenix Defence Pty. Ltd.

FIG. 12